REMARKS

Reconsideration and allowance are requested.

All of the claims stand rejected under 35 U.S.C. §101 as being related to non-statutory subject matter. This rejection is respectfully traversed.

The *State Street* statutory subject matter test quoted by the Examiner is satisfied by the pending claims. Indeed, the claims recite subject matter that provide a useful, concrete, and tangible result. The Examiner does not demonstrate or explain how the apparatus, for example, that is recited in claim 1 does not provide a useful, concrete, and tangible result or that it "consists solely of the manipulation of an abstract idea [that] is not concrete or tangibles [sic]." The burden of proof for making and supporting rejections is on the Patent Office. See *In re Piasecki*, 223 USPQ 785, 788 (Fed. Cir. 1984).

To the contrary, the independent claims in this case solve a very practical and real world technical problem that exists when designing data processing systems: how to perform co-verification of hardware and software that may be present for example in a system on chip (SoC) data processing product. In other words, the claims provide a way to test both the software and the hardware designed for an SoC to make sure it works as intended. Such diagnostic and testing tool is extremely useful to a SoC designer and/or tester. The tool is not just an abstract idea, but instead is something SoC designers and/or testers use in their regular work. The concrete and tangible result of using this co-verification tool is that SoC designs having both hardware and software elements can be thoroughly tested and any problems quickly identified and resolved before the design is to be implemented and/or produced. The tool saves money, time, and improves the quality of the resulting SoC design.

A hardware simulation of the hardware blocks to be placed within the data processing system is typically produced to allow that hardware design to be subjected to significant amounts of validation and testing in order to ensure the hardware design works correctly before being released for manufacture. Although a number of techniques have been developed for performing verification testing in respect of the hardware, it is difficult to establish co-ordination between the software routines executed by such a system and the hardware verification testing. This techniques have certain drawbacks, such as not scaling well with SoC designs that incorporate more than one processor, not enabling direct co-ordination between stimulus input from the hardware simulator and software execution state, etc. See page 4, line 22, to page 5, line 4.

In a typical co-simulation environment, the test procedure very much depends on the software code being run. In particular, the software code executes and interacts with the hardware in its own time as dictated by the flow of execution. In order to change the test procedure with respect to the hardware, it is typically necessary to also modify and recompile the software, and hence, the testing process is very time consuming and complex.

The independent claims solve this very real problem. As set out in claim 1, a debugger controls a processing unit associated with the system under verification. That processing unit is tested by executing software routines. A debugger signal interface controller interfaces with the debugger and performs one or more test actions in response to test controlling messages issued by a test manager. These test actions cause one or more stimulus signals and one or more response signals to be issued between the debugger and the debugger signal interface controller during performance of a sequence of verification tests. The test manager can then control how the debugger controls the operation of the processing unit, and thus, effectively controls the software routines executed. As a result, the test manager can control the operation of the processing unit via the debugger signal interface controller and the debugger along with the testing of the hardware units via their signal interface controller so as to coordinate the execution of the software routines with the sequence of hardware verification tests.

Again, the claimed apparatus, method, and software product solve a very real and technical problem and significantly improve the efficiency of the process of designing a data processing system such as a System-on-Chip (SoC). Moreover, at least one or more of the signal interface controller, debugger, debugger signal interface controller, and test manager in claim 1, as well as the at least one or more of the steps in method claim 18, is implemented in a suitably programmed computer, which is tangible hardware. Accordingly, the claims clearly describe statutory subject matter.

Regarding the format of claims 35-51, the Examiner's suggested formatting for those claims has been adopted. Withdrawal of the rejection under 35 U.S.C. §101 is requested.

All claims stand rejected under 35 U.S.C. §102 as being anticipated by US 6,678,645 Rajsuman. This rejection is respectfully traversed.

As will be appreciated from the above description, distinctive claimed features include the debugger which controls operation of the processing unit that is executing software routines, and the debugger signal interface controller that interfaces with the debugger and responds to test controlling messages issued by the test manager. When performing a sequence of verification tests with respect to the system under verification, the test manager typically issues test control messages to a variety of signal interface controllers that are coupled to that system. As recited in the claims, the test manager also sends test controlling messages to the debugger signal interface controller, which causes the debugger signal interface controller to perform one or more test actions during which stimulus signals and response signals are passed between the debugger and the debugger signal interface controller during performance of the sequence of verification tests. This allows the test manager to coordinate the execution of the software routines with the sequence of verification tests via the debugger signal interface controller and the debugger to control the execution of software by the processing unit.

The Examiner refers to column 5, lines 43 to 44 in connection with the plurality of signal interface controllers recited in claim 1. Accordingly, it appears that the Examiner is equating the verification units of Figure 5 with the signal interface controllers. Figure 5 shows that each verification unit is associated with a separate design validation station (DVS) which together represent a SoC having a plurality of functional cores. See column 1, lines 6 to 8.

Having made this assumption of equivalence between the verification units of Figure 5 of Rajsuman and the claimed signal interface controllers, the Examiner is unable to point to any sections in Rajsuman that disclose the remaining features recited in independent claim 1. For example, claim 1 specifies "a *debugger* operable to <u>control operation</u> of a processing unit associated with a system under verification, the processing unit being operable to execute software routines". The Examiner refers to column 11, lines 53 to 63 for this claim element. This text describes describing timing verification of critical paths of the SoC design. If the test results obtained using the vectors in the testbench data deviate from the simulation timing results, then there is an error that needs debugging. So this text in Rajsuman simply describes standard debugging activity. There is no teaching of using a debugger to control operation of a processing unit included in the hardware being tested which is executing software routines as part of a software verification test.

Rajsuman also lacks the claimed *debugger signal interface controller* for interfacing with the debugger and performing one or more test actions. The Examiner refers to the same section in column 11 of Rajsuman. This section does not disclose the claimed debugger signal interface controller. The next paragraph of claim 1 explains that the debugger signal interface controller receives test controlling messages from the test manager which identify the test actions that it needs to perform. In response, the debugger signal interface controller performs those test actions to cause stimulus signals and response signals to pass between the debugger signal interface controller and the debugger. Those stimulus and response signals affect what software is run by the processing unit because the debugger controls operation of the processing unit. In this way, the test manager can "control the operation of the processing unit via the debugger signal interface controller and the debugger in order to co-ordinate the execution of said software routines with the sequence of verification tests."

Neither the debugger nor the debugger signal interface controller of claim 1 is disclosed in Rajsuman. As a result, coordinated execution of the claimed software routines with the sequence of verification tests made possible when using the claimed debugger and debugger interface controller is also not disclosed by Rajsuman.

The application is in condition for allowance. An early notice to that effect is requested.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: [illegible]
John R. Lastova
Reg. No. 33,149

JRL:maa
901 North Glebe Road, 11th Floor
Arlington, VA 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100